`hhh4`: An endemic-epidemic modelling framework for infectious disease counts

Michaela Paul and Sebastian Meyer*
Epidemiology, Biostatistics and Prevention Institute
University of Zurich, Zurich, Switzerland

8 February 2016

Abstract

The R package **surveillance** provides tools for the visualization, modelling and monitoring of epidemic phenomena. This vignette is concerned with the `hhh4` modelling framework for univariate and multivariate time series of infectious disease counts proposed by Held et al. (2005), and further extended by Paul et al. (2008), Paul and Held (2011), Held and Paul (2012), and Meyer and Held (2014). The implementation is illustrated using several built-in surveillance data sets. The special case of *spatio-temporal* `hhh4` models is covered in greater detail in Meyer et al. (2016, Section 5).

1 Introduction

To meet the threats of infectious diseases, many countries have established surveillance systems for the reporting of various infectious diseases. The systematic and standardized reporting at a national and regional level aims to recognize all outbreaks quickly, even when aberrant cases are dispersed in space. Traditionally, notification data, i.e. counts of cases confirmed according to a specific definition and reported daily, weekly or monthly on a regional or national level, are used for surveillance purposes.
The R-package **surveillance** provides functionality for the retrospective modelling and prospective aberration detection in the resulting surveillance time series. Overviews of the outbreak detection functionality of **surveillance** are given by Höhle and Mazick (2010) and Salmon et al. (2016). This document illustrates the functionality of the function `hhh4` for the modelling of univariate and multivariate time series of infectious disease counts. It is part of the **surveillance** package as of version 1.3.

*Author of correspondence: `Sebastian.Meyer@ifspm.uzh.ch`

The remainder of this vignette unfolds as follows: Section 2 introduces the S4 class data structure used to store surveillance time series data within the package. Access and visualization methods are outlined by means of built-in data sets. In Section 3, the statistical modelling approach by Held et al. (2005) and further model extensions are described. After the general function call and arguments are shown, the detailed usage of `hhh4` is demonstrated in Section 4 using data introduced in Section 2.

2 Surveillance data

Denote by $\{y_{it}; i = 1, \ldots, I, t = 1, \ldots, T\}$ the multivariate time series of disease counts for a specific partition of gender, age and location. Here, T denotes the length of the time series and I denotes the number of units (e.g geographical regions or age groups) being monitored. Such data are represented using objects of the S4 class `sts` (surveillance time series).

The `sts` data class

The `sts` class contains the $T \times I$ matrix of counts y_{it} in a slot `observed`. An integer slot `epoch` denotes the time index $1 \leq t \leq T$ of each row in `observed`. The number of observations per year, e.g. 52 for weekly or 12 for monthly data, is denoted by `freq`. Furthermore, `start` denotes a vector of length two containing the start of the time series as `c(year, epoch)`. For spatially stratified time series, the slot `neighbourhood` denotes an $I \times I$ adjacency matrix with elements 1 if two regions are neighbors and 0 otherwise. For map visualizations, the slot `map` links the multivariate time series to geographical regions stored in a `"SpatialPolygons"` object (package **sp**). Additionally, the slot `populationFrac` contains a $T \times I$ matrix representing population fractions in unit i at time t.
The `sts` data class is also described in Höhle and Mazick (2010, Section 2.1), Salmon et al. (2016, Section 1.1), Meyer et al. (2016, Section 5.2), and on the associated help page `help("sts")`.

Some example data sets

The package **surveillance** contains a number of time series in the `data` directory. Most data sets originate from the SurvStat@RKI database[1], maintained by the Robert Koch Institute (RKI) in Germany. Selected data sets will be analyzed in Section 4 and are introduced in the following.
Note that many of the built-in datasets are stored in the S3 class data structure `disProg` used in ancient versions of the **surveillance** package (until 2006). They can be easily converted into the new S4 `sts` data structure

[1] `https://survstat.rki.de`

using the function `disProg2sts`. The resulting `sts` object can be accessed similar as standard `matrix` objects and allows easy temporal and spatial aggregation as will be shown in the remainder of this section.

Example: Influenza and meningococcal disease, Germany, 2001–2006

As a first example, the weekly number of influenza and meningococcal disease cases in Germany is considered.

```
> # load data
> data("influMen")
> # convert to sts class and print basic information about the time series
> print(fluMen <- disProg2sts(influMen))
```

```
-- An object of class sts --
freq:                 52
start:                 2001 1
dim(observed):         312 2

Head of observed:
     influenza meningococcus
[1,]         7             4

head of neighbourhood:
          influenza meningococcus
influenza         0             1
```

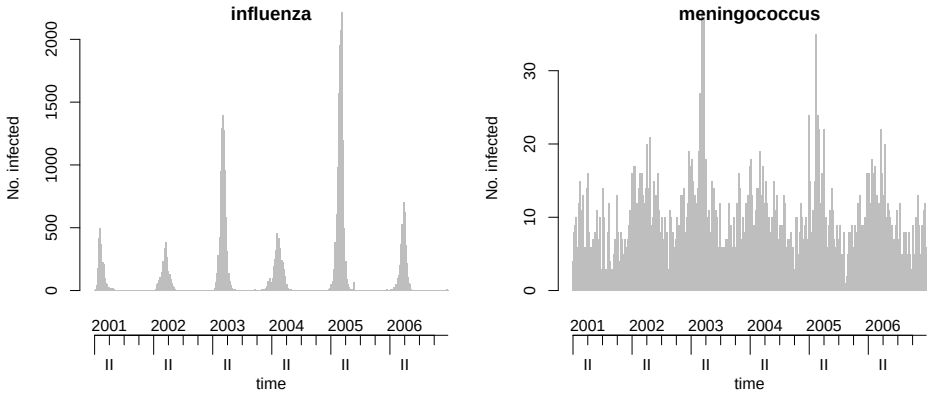
The univariate time series of meningococcal disease counts can be obtained with

```
> meningo <- fluMen[, "meningococcus"]
> dim(meningo)
```

```
[1] 312   1
```

The `plot` function provides ways to visualize the multivariate time series in time, space and space-time, as controlled by the `type` argument:

```
> plot(fluMen, type = observed ~ time | unit, # type of plot (default)
+              same.scale = FALSE,            # unit-specific ylim?
+              col = "grey")                  # color of bars
```

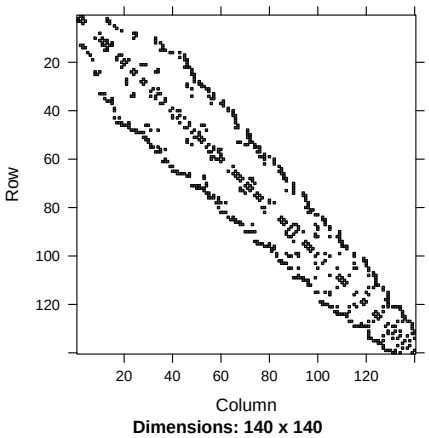


See `help("stsplot")` for a detailed description of the plot routines.

Example: Influenza, Southern Germany, 2001–2008

The spatio-temporal spread of influenza in the 140 Kreise (districts) of Bavaria and Baden-Württemberg is analyzed using the weekly number of cases reported to the RKI (Robert Koch-Institut, 2009) in the years 2001–2008. An `sts` object containing the data is created as follows:

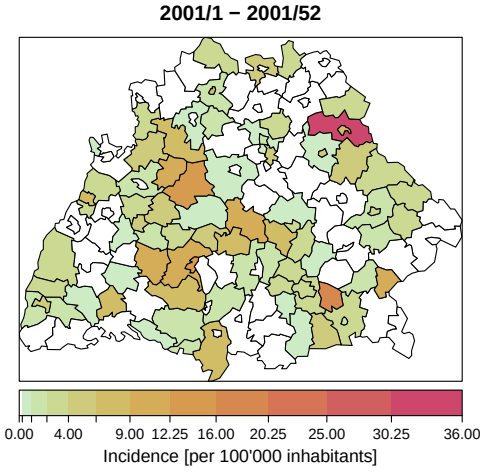
```
> # read in observed number of cases
> flu.counts <- as.matrix(read.table(system.file("extdata/counts_flu_BYBW.txt",
+                                                package = "surveillance"),
+                                    check.names = FALSE))


> # read in 0/1 adjacency matrix (1 if regions share a common border)
> nhood <- as.matrix(read.table(system.file("extdata/neighbourhood_BYBW.txt",
+                                           package = "surveillance"),
+                               check.names = FALSE))
> library("Matrix")
> print(image(Matrix(nhood)))
```



```
> # read in population fractions
> popfracs <- read.table(system.file("extdata/population_2001-12-31_BYBW.txt",
+                                    package = "surveillance"),
+                        header = TRUE)$popFrac
> # create sts object
> flu <- sts(observed = flu.counts, start = c(2001, 1), frequency = 52,
+            neighbourhood = nhood, population = popfracs)
```

These data are already included as `data("fluBYBW")` in **surveillance**. In addition to the `sts` object created above, `fluBYBW` contains a map of the administrative districts of Bavaria and Baden-Württemberg. This works by specifying a `"SpatialPolygons"` representation of the districts as an extra argument `map` in the above `sts` call. Such a `"SpatialPolygons"` object can be obtained from, e.g, an external shapefile using the function `readShapePoly` from package **maptools**. A map enables plots and animations of the cumulative number of cases by region. For instance, a disease incidence map of the year 2001 can be obtained as follows:

```
> data("fluBYBW")
> plot(fluBYBW[year(fluBYBW) == 2001, ], # select year 2001
+      type = observed ~ unit,           # total counts by region
+      population = fluBYBW@map$X31_12_01 / 100000) # per 100000 inhabitants
> grid::grid.text("Incidence [per 100'000 inhabitants]", x = 0.5, y = 0.02)
```

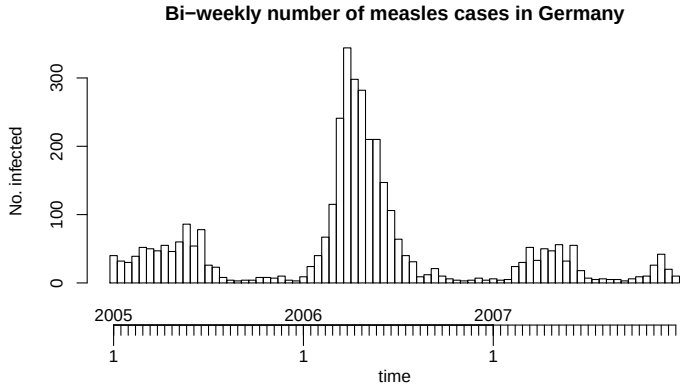


Example: Measles, Germany, 2005–2007

The following data set contains the weekly number of measles cases in the 16 German federal states, in the years 2005–2007. These data have been analyzed by Herzog et al. (2011) after aggregation into bi-weekly periods.

```
> data("measlesDE")
> measles2w <- aggregate(measlesDE, nfreq = 26)
```

```
> plot(measles2w, type = observed ~ time,  # aggregate counts over all units
+      main = "Bi-weekly number of measles cases in Germany")
```



3 Model formulation

Retrospective surveillance aims to identify outbreaks and (spatio-)temporal patterns through statistical modelling. Motivated by a branching process with immigration, Held et al. (2005) suggest the following model for the analysis of univariate time series of infectious disease counts $\{y_t; t = 1, \ldots, T\}$. The counts are assumed to be Poisson distributed with conditional mean

$$\mu_t = \lambda y_{t-1} + \nu_t, \quad (\lambda, \nu_t > 0)$$

where λ and ν_t are unknown quantities. The mean incidence is decomposed additively into two components: an epidemic or *autoregressive* component λy_{t-1}, and an *endemic* component ν_t. The former should be able to capture occasional outbreaks whereas the latter explains a baseline rate of cases with stable temporal pattern. Held et al. (2005) suggest the following parametric model for the endemic component:

$$\log(\nu_t) = \alpha + \beta t + \left\{ \sum_{s=1}^{S} \gamma_s \sin(\omega_s t) + \delta_s \cos(\omega_s t) \right\}, \tag{1}$$

where α is an intercept, β is a trend parameter, and the terms in curly brackets are used to model seasonal variation. Here, γ_s and δ_s are unknown parameters, S denotes the number of harmonics to include, and $\omega_s = 2\pi s/\texttt{freq}$ are Fourier frequencies (e.g. `freq = 52` for weekly data). For ease of interpretation, the seasonal terms in (1) can be written equivalently as

$$\gamma_s \sin(\omega_s t) + \delta_s \cos(\omega_s t) = A_s \sin(\omega_s t + \varphi_s)$$

with amplitude $A_s = \sqrt{\gamma_s^2 + \delta_s^2}$ describing the magnitude, and phase difference $\tan(\varphi_s) = \delta_s/\gamma_s$ describing the onset of the sine wave.

To account for overdispersion, the Poisson model may be replaced by a negative binomial model. Then, the conditional mean μ_t remains the same but the conditional variance increases to $\mu_t(1 + \mu_t\psi)$ with additional unknown overdispersion parameter $\psi > 0$.
The model is extended to multivariate time series $\{y_{it}\}$ in Held et al. (2005) and Paul et al. (2008) by including an additional *neighbor-driven* component, where past cases in other (neighboring) units also enter as explanatory covariates. The conditional mean μ_{it} is then given by

$$\mu_{it} = \lambda y_{i,t-1} + \phi \sum_{j \neq i} w_{ji} y_{j,t-1} + e_{it}\nu_t, \tag{2}$$

where the unknown parameter ϕ quantifies the influence of other units j on unit i, w_{ji} are weights reflecting between-unit transmission and e_{it} corresponds to an offset (such as population fractions at time t in region i). A simple choice for the weights is $w_{ji} = 1$ if units j and i are adjacent and 0 otherwise. See Paul et al. (2008) for a discussion of alternative weights, and Meyer and Held (2014) for how to estimate these weights in the spatial setting using a parametric power-law formulation based on the order of adjacency.
When analyzing a specific disease observed in, say, multiple regions or several pathogens (such as influenza and meningococcal disease), the assumption of equal incidence levels or disease transmission across units is questionable. To address such heterogeneity, the unknown quantities λ, ϕ, and ν_t in (2) may also depend on unit i. This can be done via

- unit-specific fixed parameters, e.g. $\log(\lambda_i) = \alpha_i$ (Paul et al., 2008);
- unit-specific random effects, e.g $\log(\lambda_i) = \alpha_0 + a_i$, $a_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2_\lambda)$ (Paul and Held, 2011);
- linking parameters with known (possibly time-varying) explanatory variables, e.g. $\log(\lambda_i) = \alpha_0 + x_i\alpha_1$ with region-specific vaccination coverage x_i (Herzog et al., 2011).

In general, the parameters of all three model components may depend on both time and unit. A call to `hhh4` fits a Poisson or negative binomial model with conditional mean

$$\mu_{it} = \lambda_{it} y_{i,t-1} + \phi_{it} \sum_{j \neq i} w_{ji} y_{j,t-1} + e_{it}\nu_{it}$$

to a (multivariate) time series of counts. Here, the three unknown quantities are modelled as log-linear predictors

$$\log(\lambda_{it}) = \alpha_0 + a_i + \boldsymbol{u}_{it}^\top \boldsymbol{\alpha} \tag{ar}$$

$$\log(\phi_{it}) = \beta_0 + b_i + \boldsymbol{x}_{it}^\top \boldsymbol{\beta} \tag{ne}$$

$$\log(\nu_{it}) = \gamma_0 + c_i + \boldsymbol{z}_{it}^\top \boldsymbol{\gamma} \tag{end}$$

where $\alpha_0, \beta_0, \gamma_0$ are intercepts, $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ are vectors of unknown parameters corresponding to covariate vectors $\boldsymbol{u}_{it}, \boldsymbol{x}_{it}, \boldsymbol{z}_{it}$, and a_i, b_i, c_i are random effects. For instance, model (1) with $S = 1$ seasonal terms may be represented as $\boldsymbol{z}_{it} = (t, \sin(2\pi/\texttt{freq}\, t), \cos(2\pi/\texttt{freq}\, t))^\top$. The stacked vector of all random effects is assumed to follow a normal distribution with mean $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}$. In applications, each of the components `ar`, `ne`, and `end` may be omitted in parts or as a whole.
If the model does not contain random effects, standard likelihood inference can be performed. Otherwise, inference is based on penalized quasi-likelihood as described in detail in Paul and Held (2011).

4 Function call and control settings

The estimation procedure is called with

```
> hhh4(sts, control)
```

where `sts` denotes a (multivariate) surveillance time series and the model is specified in the argument `control` in consistency with other algorithms in **surveillance**. The `control` setting is a list of the following arguments (here with default values):

```
> control = list(
+     ar = list(f = ~ -1,               # formula for log(lambda_it)
+               offset = 1),            # optional multiplicative offset
+     ne = list(f = ~ -1,               # formula for log(phi_it)
+               offset = 1,             # optional multiplicative offset
+               weights = neighbourhood(stsObj) == 1),  # (w_ji) matrix
+     end = list(f = ~ 1,               # formula for log(nu_it)
+                offset = 1),           # optional multiplicative offset e_it
+     family = "Poisson",               # Poisson or NegBin model
+     subset = 2:nrow(stsObj),          # subset of observations to be used
+     optimizer = list(stop = list(tol = 1e-5, niter = 100), # stop rules
+                      regression = list(method = "nlminb"), # for penLogLik
+                      variance = list(method = "nlminb")),  # for marLogLik
+     verbose = FALSE,                  # level of progress reporting
+     start = list(fixed = NULL,        # list with initial values for fixed,
+                  random = NULL,       # random, and
+                  sd.corr = NULL),     # variance parameters
+     data = list(t = epoch(stsObj)-1),# named list of covariates
+     keep.terms = FALSE                # whether to keep the model terms
+ )
```

The first three arguments `ar`, `ne`, and `end` specify the model components using `formula` objects. By default, the counts y_{it} are assumed to be Poisson distributed, but a negative binomial model can be chosen by setting `family = "NegBin1"`. By default, both the penalized and marginal log-likelihoods are maximized using the quasi-Newton algorithm available via the R function `nlminb`. The methods from `optim` may also be used, e.g.,

`optimizer = list(variance = list(method="Nelder-Mead")` is a useful alternative for maximization of the marginal log-likelihood with respect to the variance parameters. Initial values for the fixed, random, and variance parameters can be specified in the `start` argument. If the model contains covariates, these have to be provided in the `data` argument. If a covariate does not vary across units, it may be given as a vector of length T. Otherwise, covariate values must be given in a matrix of size $T \times I$.
In the following, the functionality of `hhh4` is demonstrated using the data sets introduced in Section 2 and previously analyzed in Paul et al. (2008), Paul and Held (2011) and Herzog et al. (2011). Selected results are reproduced. For a thorough discussion we refer to these papers.

Univariate modelling

As a first example, consider the univariate time series of meningococcal infections in Germany, 01/2001–52/2006 (cf. Paul et al., 2008, Table 1). A Poisson model without autoregression and $S = 1$ seasonal term is specified as follows:

```
> # specify a formula object for the endemic component
> ( f_S1 <- addSeason2formula(f = ~ 1, S = 1, period = 52) )


~1 + sin(2 * pi * t/52) + cos(2 * pi * t/52)


> # fit the Poisson model
> result0 <- hhh4(meningo, control = list(end = list(f = f_S1),
+                                         family = "Poisson"))
> summary(result0)


Call:
hhh4(stsObj = meningo, control = list(end = list(f = f_S1), family = "Poisson"))

Coefficients:
                      Estimate  Std. Error
end.1                 2.26478   0.01871
end.sin(2 * pi * t/52)  0.36195   0.02590
end.cos(2 * pi * t/52)  0.26055   0.02578

Log-likelihood:   -872.09
AIC:              1750.19
BIC:              1761.41

Number of units:        1
Number of time points:  311
```

To fit the corresponding negative binomial model, we can use the convenient `update` method:

```
> result1 <- update(result0, family = "NegBin1")
```

Note that the `update` method by default uses the parameter estimates from the original model as start values when fitting the updated model; see `help("update.hhh4")` for details.
We can calculate Akaike's Information Criterion for the two models to check whether accounting for overdispersion is useful for these data:

```
> AIC(result0, result1)
```

```
        df      AIC
result0  3 1750.187
result1  4 1708.344
```

Due to the default control settings with `ar = list(f = ~ -1)`, the autoregressive component has been omitted in the above models. It can be included by the following model update:

```
> # fit an autoregressive model
> result2 <- update(result1, ar = list(f = ~ 1))
```

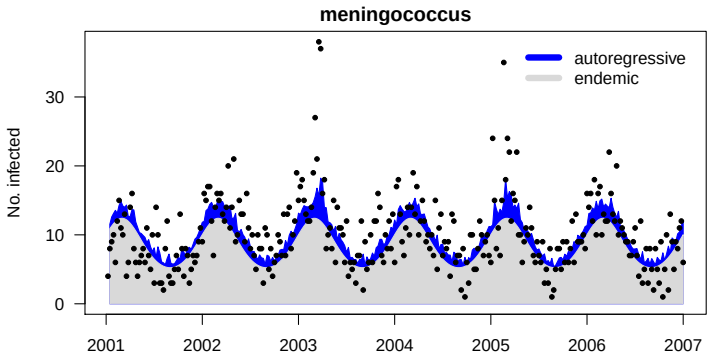
To extract only the ML estimates and standard errors instead of a full model `summary`, the `coef` method can be used:

```
> coef(result2, se = TRUE,    # also return standard errors
+      amplitudeShift = TRUE, # transform sine/cosine coefficients
+                             # to amplitude/shift parameters
+      idx2Exp = TRUE)        # exponentiate remaining parameters
```

```
                       Estimate Std. Error
exp(ar.1)            0.16471293 0.05513307
exp(end.1)           8.05551496 0.55270775
end.A(2 * pi * t/52) 0.43157655 0.03793440
end.s(2 * pi * t/52) 0.65109457 0.04841228
overdisp             0.04917299 0.01168151
```

Here, `exp(ar.1)` is the autoregressive coefficient λ and can be interpreted as the epidemic proportion of disease incidence (Held and Paul, 2012). Note that the above transformation arguments `amplitudeShift` and `idx2Exp` can also be used in the `summary` method. Many other standard methods are implemented for `"hhh4"` fits, see, e.g., `help("confint.hhh4")`.
A plot of the fitted model components can be easily obtained:

```
> plot(result2)
```



See the comprehensive `help("plot.hhh4")` for further options.

Bivariate modelling

Now, the weekly numbers of both meningococcal disease (MEN) and influenza (FLU) cases are analyzed to investigate whether influenza infections predispose meningococcal disease (cf. Paul et al., 2008, Table 2). This requires disease-specific parameters which are specified in the formula object with `fe(...)`. In the following, a negative binomial model with mean

$$\begin{pmatrix} \mu_{\text{men},t} \\ \mu_{\text{flu},t} \end{pmatrix} = \begin{pmatrix} \lambda_{\text{men}} & \phi \\ 0 & \lambda_{\text{flu}} \end{pmatrix} \begin{pmatrix} \text{MEN}_{t-1} \\ \text{FLU}_{t-1} \end{pmatrix} + \begin{pmatrix} \nu_{\text{men},t} \\ \nu_{\text{flu},t} \end{pmatrix},$$

where the endemic component includes $S = 3$ seasonal terms for the FLU data and $S = 1$ seasonal terms for the MEN data is considered. Here, ϕ quantifies the influence of past influenza cases on the meningococcal disease incidence. This model corresponds to the second model of Table 2 in Paul et al. (2008) and is fitted as follows:

```
> # no "transmission" from meningococcus to influenza
> neighbourhood(fluMen)["meningococcus","influenza"] <- 0
> neighbourhood(fluMen)
```

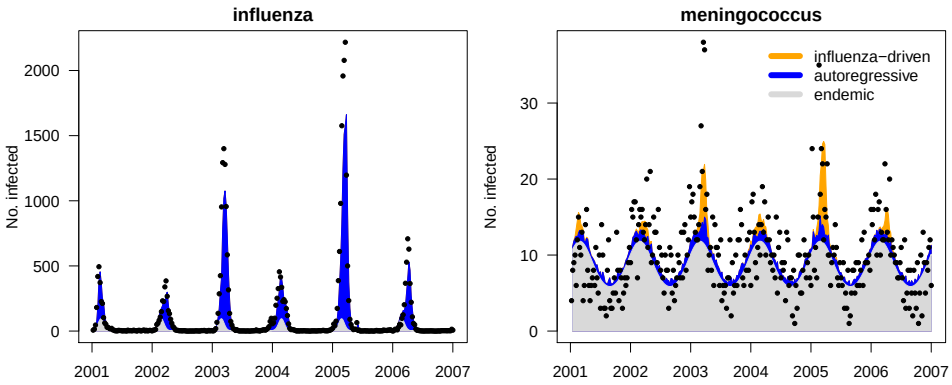
```
             influenza meningococcus
influenza            0             1
meningococcus        0             0
```

```
> # create formula for endemic component
> f.end <- addSeason2formula(f = ~ -1 + fe(1, unitSpecific = TRUE),
+                                          # disease-specific intercepts
+                            S = c(3, 1),    # S = 3 for flu, S = 1 for men
+                            period = 52)
> # specify model
> m <- list(ar = list(f = ~ -1 + fe(1, unitSpecific = TRUE)),
+           ne = list(f = ~ 1,  # phi, only relevant for meningococcus due to
```

```
+                     weights = neighbourhood(fluMen)),   # the weight matrix
+           end = list(f = f.end),
+           family = "NegBinM") # disease-specific overdispersion
> # fit model
> result <- hhh4(fluMen, control = m)
> summary(result, idx2Exp=1:3)


Call:
hhh4(stsObj = fluMen, control = m)

Coefficients:
                                        Estimate   Std. Error
exp(ar.1.influenza)                     0.737592   0.050030
exp(ar.1.meningococcus)                 0.095146   0.056894
exp(ne.1)                               0.005425   0.001413
end.1.influenza                         1.088286   0.165319
end.1.meningococcus                     2.118598   0.066831
end.sin(2 * pi * t/52).influenza        1.186185   0.235984
end.sin(2 * pi * t/52).meningococcus    0.266606   0.039680
end.cos(2 * pi * t/52).influenza        1.509778   0.146707
end.cos(2 * pi * t/52).meningococcus    0.229044   0.035323
end.sin(4 * pi * t/52).influenza        0.919169   0.171502
end.cos(4 * pi * t/52).influenza       -0.161599   0.179850
end.sin(6 * pi * t/52).influenza        0.369235   0.149980
end.cos(6 * pi * t/52).influenza       -0.534546   0.161901
overdisp.influenza                      0.294554   0.035769
overdisp.meningococcus                  0.039497   0.010893

Log-likelihood:   -1880.97
AIC:              3791.94
BIC:              3858.43

Number of units:        2
Number of time points:  311
```

A plot of the estimated mean components can be obtained as follows:

```
> plot(result, units = 1:2, legend = 2, legend.args = list(
+      legend = c("influenza-driven", "autoregressive", "endemic")))
```



Multivariate modelling

For disease counts observed in a large number of regions, say, (i.e. highly multivariate time series of counts) the use of region-specific parameters to account for regional heterogeneity is no longer feasible as estimation and identifiability problems may occur. Here we illustrate two approaches: region-specific random effects and region-specific covariates.

Influenza, Southern Germany, 2001–2008

Paul and Held (2011) propose a random effects formulation to analyze the weekly number of influenza cases in 140 districts of Southern Germany. For example, consider a model with random intercepts in the endemic component: $c_i \overset{iid}{\sim} \mathcal{N}(0, \sigma_\nu^2), i = 1, \ldots, I$. Such effects are specified as:

```
> f.end <- ~ -1 + ri(type = "iid", corr = "all")
```

The alternative `type = "car"` would assume spatially correlated random effects; see Paul and Held (2011) for details. The argument `corr = "all"` allows for correlation between region-specific random effects in different components, e.g., random incidence levels c_i in the endemic component and random effects b_i in the neighbor-driven component. The following call to `hhh4` fits such a random effects model with linear trend and $S = 3$ seasonal terms in the endemic component, a fixed autoregressive parameter λ, and first-order transmission weights $w_{ji} = \mathbb{I}(j \sim i)$ – normalized such that $\sum_i w_{ji} = 1$ for all rows j – to the influenza data (cf. Paul and Held, 2011, Table 3, model B2).

```
> # endemic component: iid random effects, linear trend, S=3 seasonal terms
> f.end <- addSeason2formula(f = ~ -1 + ri(type="iid", corr="all") +
+                                I((t-208)/100),
+                            S = 3, period = 52)
> # model specification
> model.B2 <- list(ar = list(f = ~ 1),
+                  ne = list(f = ~ -1 + ri(type="iid", corr="all"),
+                            weights = neighbourhood(fluBYBW),
+                            normalize = TRUE),  # all(rowSums(weights) == 1)
+                  end = list(f = f.end, offset = population(fluBYBW)),
+                  family = "NegBin1", verbose = TRUE,
+                  optimizer = list(variance = list(method = "Nelder-Mead")))
> # default start values for random effects are sampled from a normal
> set.seed(42)


> # fit the model (takes about 35 seconds)
> result.B2 <- hhh4(fluBYBW, model.B2)
> summary(result.B2, maxEV = TRUE, idx2Exp = 1:3)
```

```
Call:
hhh4(stsObj = fluBYBW, control = model.B2)

Random effects:
            Var    Corr
ne.ri(iid)  0.9642
end.ri(iid) 0.5067 0.5652

Fixed effects:
                           Estimate  Std. Error
exp(ar.1)                   0.40986   0.01507
exp(ne.ri(iid))             0.21927   0.02269
exp(end.I((t - 208)/100))   1.77478   0.04199
end.sin(2 * pi * t/52)      2.17871   0.09808
end.cos(2 * pi * t/52)      2.33738   0.12134
end.sin(4 * pi * t/52)      0.45161   0.10427
end.cos(4 * pi * t/52)     -0.37668   0.09404
end.sin(6 * pi * t/52)      0.30145   0.06452
end.cos(6 * pi * t/52)     -0.24798   0.06294
end.ri(iid)                 0.22344   0.10222
overdisp                    1.08439   0.03392

Epidemic dominant eigenvalue:  0.71

Penalized log-likelihood:  -18696.6
Marginal log-likelihood:   -343.59

Number of units:        140
Number of time points:  415
```

Model choice based on information criteria such as AIC or BIC is well explored and understood for models that correspond to fixed-effects likelihoods. However, in the presence of random effects their use can be problematic. For model selection in time series models, the comparison of successive one-step-ahead forecasts with the actually observed data provides a natural alternative. In this context, Gneiting and Raftery (2007) recommend the use of strictly proper scoring rules, such as the logarithmic score (logs) or the ranked probability score (rps). See Czado et al. (2009) and Paul and Held (2011) for further details.
One-step-ahead predictions for the last 2 years for model B2 could be obtained as follows:

```
> pred.B2 <- oneStepAhead(result.B2, tp = nrow(fluBYBW) - 2*52)
```

However, computing "rolling" one-step-ahead predictions from a random effects model is computationally expensive, since the model needs to be refitted at every time point. The above call would take approximately 45 minutes! So for the purpose of this vignette, we use the fitted model based on the whole time series to compute all (fake) predictions during the last two years:

```
> predfinal.B2 <- oneStepAhead(result.B2, tp = nrow(fluBYBW) - 2*52,
+                              type = "final")
```

The mean scores (logs and rps) corresponding to this set of predictions can then be computed as follows:

```
> colMeans(scores(predfinal.B2, which = c("logs", "rps")))
```

```
     logs       rps
0.5430243 0.4165988
```

Using predictive model assessments, Meyer and Held (2014) found that power-law transmission weights more appropriately reflect the spread of influenza than the previously used first-order weights (which actually allow the epidemic to spread only to directly adjacent districts within one week). These power-law weights can be constructed by the function `W_powerlaw` and require the `neighbourhood` of the `sts` object to contain adjacency orders. The latter can be easily obtained from the binary adjacency matrix using the function `nbOrder`. See the corresponding help pages or Meyer et al. (2016, Section 5) for illustrations.

Measles, German federal states, 2005–2007

As a last example, consider the number of measles cases in the 16 federal states of Germany, in the years 2005–2007. There is considerable regional variation in the incidence pattern which is most likely due to differences in vaccination coverage. In the following, information about vaccination coverage in each state, namely the log proportion of unvaccinated school starters, is included as explanatory variable in a model for the bi-weekly aggregated measles data. See Herzog et al. (2011) for further details.
Vaccination coverage levels for the year 2006 are available in the dataset `data(MMRcoverageDE)`. This dataset can be used to compute the 78×16 matrix `vac0` with adjusted proportions of unvaccinated school starters in each state i used by Herzog et al. (2011). The first few entries of this matrix are shown below:

```
> vac0[1:2, 1:6]
```

```
     Baden-Wuerttemberg  Bavaria   Berlin Brandenburg   Bremen   Hamburg
[1,]          0.1000115 0.113261 0.099989   0.0605575 0.115963 0.0999685
[2,]          0.1000115 0.113261 0.099989   0.0605575 0.115963 0.0999685
```

We fit a Poisson model, which links the autoregressive parameter with this covariate and contains $S = 1$ seasonal term in the endemic component (cf. Herzog et al., 2011, Table 3, model A0):

```
> # endemic component: Intercept + sine/cosine terms
> f.end <- addSeason2formula(f = ~ 1, S = 1, period = 26)
> # autoregressive component: Intercept + vaccination coverage information
> model.A0 <- list(ar = list(f = ~ 1 + logVac0),
+                  end = list(f = f.end, offset = population(measles2w)),
+                  data = list(t = epoch(measles2w), logVac0 = log(vac0)))
> # fit the model
> result.A0 <- hhh4(measles2w, model.A0)
> summary(result.A0, amplitudeShift = TRUE)
```

```
Call:
hhh4(stsObj = measles2w, control = model.A0)

Coefficients:
                      Estimate  Std. Error
ar.1                   3.00578   0.51652
ar.logVac0             1.38314   0.22642
end.1                  1.77568   0.06031
end.A(2 * pi * t/26)   0.66021   0.08411
end.s(2 * pi * t/26)  -0.09829   0.12184

Log-likelihood:   -1778.06
AIC:              3566.12
BIC:              3591.71

Number of units:        16
Number of time points:  77
```

5 Conclusion

As part of the R package **surveillance**, the function `hhh4` provides a flexible tool for the modelling of multivariate time series of infectious disease counts. The presented count data model is able to account for serial and spatio-temporal correlation, as well as heterogeneity in incidence levels and disease transmission.

References

Czado, C., Gneiting, T., and Held, L. (2009). Predictive model assessment for count data. *Biometrics*, 65(4):1254–1261.

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378.

Held, L., Höhle, M., and Hofmann, M. (2005). A statistical framework for the analysis of multivariate infectious disease surveillance counts. *Statistical Modelling*, 5(3):187–199.

Held, L. and Paul, M. (2012). Modeling seasonality in space-time infectious disease surveillance data. *Biometrical Journal*, 54(6):824–843.

Herzog, S. A., Paul, M., and Held, L. (2011). Heterogeneity in vaccination coverage explains the size and occurrence of measles epidemics in German surveillance data. *Epidemiology and Infection*, 139:505–515.

Höhle, M. and Mazick, A. (2010). Aberration detection in R illustrated by Danish mortality monitoring. In Kass-Hout, T. and Zhang, X., editors, *Biosurveillance: A Health Protection Priority*. CRC Press.

Meyer, S. and Held, L. (2014). Power-law models for infectious disease spread. *Annals of Applied Statistics*, 8(3):1612–1639.

Meyer, S., Held, L., and Höhle, M. (2016). Spatio-temporal analysis of epidemic phenomena using the R package `surveillance`. *Journal of Statistical Software*. Preprint available at `http://arxiv.org/abs/1411.0416`.

Paul, M. and Held, L. (2011). Predictive assessment of a non-linear random effects model for multivariate time series of infectious disease counts. *Statistics in Medicine*, 30(10):1118–1136.

Paul, M., Held, L., and Toschke, A. M. (2008). Multivariate modelling of infectious disease surveillance data. *Statistics in Medicine*, 27(29):6250–6267.

Robert Koch-Institut (2009). SurvStat@RKI. `http://www3.rki.de/SurvStat`. Accessed March 2009.

Salmon, M., Schumacher, D., and Höhle, M. (2016). Monitoring count time series in R: Aberration detection in public health surveillance. *Journal of Statistical Software*. Preprint available at `http://arxiv.org/abs/1411.1292`.